

Biava, M., Woodgate, M., and Barakos, G. N. (2016) Fully implicit discrete-adjoint methods for rotorcraft applications. AIAA Journal, (doi:10.2514/1.J054006)

There may be differences between this version and the published version. You are advised to consult the publisher's version if you wish to cite from it.

<http://eprints.gla.ac.uk/116456/>

Deposited on: 26 February 2016

Fully Implicit Discrete Adjoint Methods for Rotorcraft Applications

Massimo Biava¹, Mark Woodgate² and George N. Barakos³

ABSTRACT

This paper presents the development of a fully implicit, low-memory, discrete adjoint method by means of automatic source-code differentiation applied to the Helicopter Multi-Block computational fluid dynamics solver. The method is suitable for applications in flight mechanics as well as shape optimization, and is demonstrated in this paper for popular flow cases reported in the literature. In particular, adjoint CFD computations were undertaken for airfoils, wings and rotor blade cases, and the obtained results were found to agree well with published solutions and with finite differences of flow derivatives. The method has been demonstrated for inviscid and viscous cases and results suggest that the current implementation is robust and efficient. The cost of the adjoint computations is relatively low due to the employed source-code differentiation and most of the times it is no more than the cost of a steady-state flow solution.

NOMENCLATURE

a	=	lift slope factor [Eq. (31)]
C_D	=	drag coefficient
C_L	=	lift coefficient
C_l	=	roll moment coefficient
C_M	=	airfoil pitch moment coefficient
C_m	=	pitch moment coefficient
C_n	=	yaw moment coefficient
C_Q	=	rotor torque coefficient
C_T	=	rotor thrust coefficient
C_Y	=	side force coefficient
c	=	chord [Eq. (31)]
D	=	matrix in harmonic-balance equation [Eq. (13)]
I	=	functional of flow solution
J	=	Jacobian matrix
k	=	reduced frequency [Eq. (30)]
M_∞	=	freestream Mach number
\mathbf{N}	=	mesh-metric vector
N_H	=	number of harmonics
\mathbf{P}	=	flow solution vector, primitive variables
p	=	roll rate
q	=	pitch rate
\mathbf{R}	=	flow equation residual vector
Re	=	Reynolds number
r	=	yaw rate
U_∞	=	freestream velocity [Eq. (31)]
\mathbf{W}	=	flow-solution vector, conservative variables
w	=	rotor axial velocity
\mathbf{X}	=	mesh coordinate vector
$\dot{\mathbf{X}}$	=	mesh velocity vector
x	=	independent variable
α	=	pitch angle
β	=	sideslip angle, blade-flap angle
θ	=	rotor collective pitch
θ_{tw}	=	blade twist angle

¹Research Associate, University of Glasgow, James Watt South Building, Glasgow G12 8QQ, U.K.

²Research Associate, University of Glasgow, James Watt South Building, Glasgow G12 8QQ, U.K.

³Professor, University of Glasgow, James Watt South Building, Glasgow G12 8QQ, U.K., corresponding author.

λ	=	adjoint-variable vector [Eq. (4)]
σ	=	rotor solidity [Eq. (31)]
ω	=	specific turbulence dissipation, periodic-flow frequency

Subscript

hb	=	harmonic balance
i,j,k	=	mesh-cell indices
∞	=	freestream value

Superscript

n	=	time level
-----	---	------------

1 INTRODUCTION

The design of new generation helicopters with increased performance and improved handling qualities requires a deeper understanding of their aerodynamics, not only in steady flight, but also during manoeuvres. Because of the nonlinearity and unsteadiness of rotor flows, it is extremely challenging to understand their aerodynamic characteristics. To reduce the complexity of the problem, it is commonly assumed that for small deviations from a given steady flight condition, the flight dynamics behaviour can be described by means of a linearized model, defined by a set of aerodynamic derivatives. These derivatives can be obtained via finite differences (FD) out of a CFD computation. Finite differencing becomes prohibitive in terms of computational cost, because two or more complete flow solutions are required to compute each derivative.

A more economic way to obtain the aerodynamic derivatives with CFD is via solving the *sensitivity equation* (2), casted in either tangent or adjoint form [1, 2]. The basic idea is to write any aerodynamic force and moment coefficient I as a function of the flow variables \mathbf{W} and of the input flight dynamics variable of interest x (angle of attack, sideslip, Mach number, etc.), that is $I = I[\mathbf{W}(x), x]$. The flow variables are subject to satisfy the fluid dynamics governing equations [*e.g.* the Navier–Stokes equations (NS)] written in compact form as

$$(1) \quad \mathbf{R}[\mathbf{W}(x), x] = 0.$$

Formally, taking the derivative of I with respect to x we obtain:

$$(2) \quad \frac{\mathcal{D}I}{\mathcal{D}x} = \frac{\partial I}{\partial x} + \frac{\partial I}{\partial \mathbf{W}} \frac{\partial \mathbf{W}}{\partial x},$$

which represents the tangent form of the sensitivity equation. All the partial derivatives appearing on the right-hand side can be computed with limited effort, with the exception of $\partial \mathbf{W} / \partial x$, that represents the variation of the flow variables with respect to the independent input parameters. This last term may be obtained by differentiating the governing equations (1), to yield the following linear system for the unknown $\partial \mathbf{W} / \partial x$:

$$(3) \quad \frac{\mathcal{D}\mathbf{R}}{\mathcal{D}x} = 0 \quad \implies \quad \frac{\partial \mathbf{R}}{\partial \mathbf{W}} \frac{\partial \mathbf{W}}{\partial x} = -\frac{\partial \mathbf{R}}{\partial x}.$$

Therefore, the computation of a flow sensitivity is reduced to the solution of the nonlinear governing flow equations (1) plus the solution of the linear system (3). This linear problem is however usually hard to compute, since the Jacobian matrix $\partial \mathbf{R} / \partial \mathbf{W}$ is characterized by a high stiffness, and the solution time is usually comparable to that of the base flow. The right-hand side of Eq. (3) depends upon the input variable x , and therefore, like with finite differencing, the computational cost for computing aerodynamic derivatives scales with the number of flight dynamics variables. However, the sensitivity equation approach requires the solution of only one linear system of equations for each derivative and does not suffer of cancellation errors, yielding results accurate up to machine precision. Instead, using second order finite differences for instance, two nonlinear problems have to be solved to compute each derivative.

The sensitivity problem (2)-(3) can be recast in dual form by introducing the adjoint variable vector λ as the solution of the following linear system [3]:

$$(4) \quad \left(\frac{\partial \mathbf{R}}{\partial \mathbf{W}} \right)^T \lambda = - \left(\frac{\partial I}{\partial \mathbf{W}} \right)^T.$$

Substituting Eq. (4) into Eq. (2) and using Eq. (3) we obtain:

$$(5) \quad \frac{\mathcal{D}I}{\mathcal{D}x} = \frac{\partial I}{\partial x} - \lambda^T \frac{\partial \mathbf{R}}{\partial \mathbf{W}} \frac{\partial \mathbf{W}}{\partial x} \quad \implies \quad \frac{\mathcal{D}I}{\mathcal{D}x} = \frac{\partial I}{\partial x} + \lambda^T \frac{\partial \mathbf{R}}{\partial x}.$$

The computational cost of the dual sensitivity problem (4)-(5) scales with the number of outputs, since the right-hand side of Eq. (4) depends on I , but it is independent of the input parameters. The choice between the use of the direct or dual sensitivity

problem consequently depends on the balance between the number of outputs and the number of inputs. The two methods, for instance, should perform equally well in computing the flight-mechanic derivatives, since the number of output force and moment coefficients is similar to the number of input flight-mechanic parameters.

The calculation of the partial derivatives appearing in the sensitivity equation can be done manually, by deriving analytical expressions, and writing the necessary computer code. Nonetheless, this approach can be tedious if the flow equations involve complex terms, like upwinding terms for the inviscid fluxes or source terms of turbulence models. Recent advances in *automatic differentiation* (AD) tools [4, 5, 6], however, enable to produce the computer code for the differentials of these complex terms directly from the source code of the CFD solver [7, 8, 9].

The present work describes the development of the sensitivity equation approach by means of AD in the CFD solver Helicopter Multi Block (HMB2) [10, 11] of Liverpool and AgustaWestland. As followed in the works of Mader *et al.* [8] and Jones *et al.* [9], the individual functions of the CFD solver have been automatically differentiated and assembled afterwards to build the necessary terms in the sensitivity equation, both in tangent and adjoint forms. The linear system associated to the sensitivity problem is solved using a matrix-free version of the fully implicit fixed-point iteration scheme of the base flow solver. The resulting code is able to compute the aerodynamic derivatives of fixed wing aircraft and of rotors, at a fraction of the cost required by finite differencing. The method is demonstrated for the aerodynamic sensitivities of the NACA0012 airfoil, the ONERA-The French Aerospace Lab M6 wing, the ONERA-The French Aerospace Lab 7AD and the S-76 rotors. Both steady and periodic flow solutions were analyzed. To our knowledge this is a first implementation of a matrix-free fully implicit adjoint solver of the NS equations with turbulence modelling included.

2 BACKGROUND ON AD AND THE DISCRETE ADJOINT

The first application of the adjoint method to fluid dynamics is dated back to 1974 with the pioneering work of Pironneau [12], in which adjoint methods and control theory were applied to drag minimization. Starting from the late eighties the first applications to CFD problems begin to appear, with the works of Jameson and his co-authors [13, 14, 15]. They exploited the adjoint method and control theory for aerodynamic shape optimization in conjunction with CFD techniques, whose complexity increased over the years from the solution of the potential flow equations to that of the NS equations [16, 17, 18]. The derivation of the adjoint problem in these works is based on the *continuous approach* (CA), where the adjoint equations are analytically derived from the primary flow equations and discretized afterwards.

The alternative *discrete approach* (DA) to the adjoint problem consists in deriving the adjoint equations directly from the discretized formulation of the flow equations. This has been pursued in the works of Elliott and Peraire [19], Anderson [20] and Mavriplis [21] in the context of aerodynamic shape optimization with unstructured meshes. A fairly complete overview of the development of continuous and discrete adjoint methods in the last two decades of the 20th century can be found in Newman *et al.* [22]. Both continuous and discrete approaches have advantages and disadvantages, as pointed out by Giles and Pierce [3]. These are summarized in table 1.

Discrete approach	Continuous approach
Provides the exact gradients, since the discrete adjoint operator is simply the transpose of the matrix arising from the discretization of the primary flow equations	Gives an approximation to the continuous gradient based on some alternative discretization
The implementation requires less coding effort, especially if AD is employed	Requires hand coding of the discretization scheme applied to the continuous adjoint equations
Straight application of AD to the CFD code produces inefficient adjoint code, so that application of AD to individual nonlinear functions and partial re-coding is necessary	The continuous code is often considerably simpler than the discrete in terms of operation count and memory requirements, as well as easier to implement
The derivation of the adjoint equations and BCs is purely algebraic, and gives no insight in the physics of the problem	Gives a more clear interpretation of the physics behind the adjoint variables and of the associated BCs

Table 1: Advantages and disadvantages of the continuous and discrete adjoint approaches.

The implementation of the DA for flow equations involving complex terms (upwinding terms, terms depending on spectral radii, source terms appearing in turbulence models, etc.) is not straightforward. A technique to tackle the problem of deriving the discrete adjoint in such complex cases is AD, in which the adjoint code to evaluate the gradients is obtained by directly

manipulating the original CFD code, as in the work of Mohammadi [23, 24]. This approach has grown in popularity and was later pursued by other research groups [7, 8, 9, 25, 26].

AD may be obtained by means of source-transformation tools or via operator overloading in programming languages such as FORTRAN 90 and C++. Tools that use source code transformation add new statements to the original source code that compute the derivatives of the original statements. The operator overloading approach consists of a new user-defined type that is used instead of floating points. This new type includes not only the value of the original variable, but its derivative as well. The operator overloading approach results in fewer changes to the original code, but is usually less efficient [4]. AD tools are available for a variety of programming languages. ADIFOR [27], TAF [28] and TAMC [29] are some of the tools available for FORTRAN. TAPENADE [30, 31] supports both FORTRAN 90 and C. A complete list of AD tools available for each programming language may be found at [32].

There are two different modes of operation for AD of a computer code: the *forward* (or *tangent*) mode and the *reverse* (or *adjoint*) mode. The forward mode uses the chain rule to propagate the required derivatives in the same direction of the original computer code. The cost of forward AD is proportional to the number of inputs of the computed function. In reverse mode, the derivatives are propagated backward, from the last statement of the code to the first. The reverse mode is analogous to the adjoint method and the cost is proportional to the number of outputs of the computed function. However, the memory requirements of the reverse mode can be considerably higher, since the storage of intermediate results of the function evaluation is required for the backward propagation of the derivatives.

It is to be noted that AD cannot be applied directly to the whole residual evaluation chain to produce the adjoint of the flow equations, because it would lead to an inefficient code in terms of memory and CPU time. A more realistic goal for AD is in assisting the derivation of the discrete adjoint by hand-differentiation, by automatically differentiating, and by transposing individual source code functions. This approach was adopted, for instance, in Mader *et al.* [8] and in Jones *et al.* [9] and was also followed here.

3 PAST WORK ON ADJOINT FOR AIRCRAFT AND ROTORCRAFT APPLICATIONS

The introduction of AD, the advances in techniques for solving the adjoint problem and the growing power of computing hardware allowed application of the adjoint method to more complex cases. Also, driven by the industrial need of more realistic flight-mechanic models, the related research widened its initial objective of aerodynamic shape optimization to make space to novel applications such as aeromechanics. In the work of Limache and Cliff [1] and of Mader and Martins [2, 33], for instance, the aerodynamic derivatives of airfoil and wings are computed by solving the sensitivity problem.

This concept was then extended to compute the sensitivities of time-periodic flow solutions, such as those generated by turbomachinery and helicopters, by applying the adjoint method to the time-spectral formulation of the flow equations, which reduces the time-dependent problem to a steady problem in the frequency domain. This is described in Choi *et al.* [34] and in Mader and Martins [33] for helicopter applications, and in Huang and Ekici [35] in the context of turbomachinery. The adjoint development in these works is, however, limited to inviscid flows.

In the context of the discrete adjoint, full account of the viscous effects and linearization of the RANS equations coupled with a turbulence model has been attempted by several authors [36, 37, 38, 39, 40]. Applications to realistic aerodynamic optimization problems can also be found in [41] for fixed wing aircraft, and in Mani and Mavriplis [42] for rotorcraft.

These past works highlighted the efficiency of the sensitivity equation approach for aerodynamic shape optimization and for aeromechanics applications. They also showed the difficulties associated to the convergence of the sensitivity equation and the demanding memory requirements of adjoint solvers, which can represent a limiting factor for realistic large-scale applications. The objective of the present work is to partially overcome these drawbacks, while keeping the efficiency and accuracy of the sensitivity problem approach for the computation of aerodynamic derivatives. The novelty of the work is the detailed description of the development of a low-memory fully implicit adjoint solver for the RANS equations, and the combination with the harmonic balance method [43, 44].

4 THE HMB2 FLOW SOLVER

The following contains a brief outline of the approach used in the Helicopter Multi-Block solver version 2.0. The NS equations are discretized using a cell-centred finite volume approach. The computational domain is divided into a finite number of non-overlapping control-volumes V_{ijk} , and the governing equations are applied to each cell. Also, the NS equations are re-written in a curvilinear co-ordinate system which simplifies the formulation of the discretized terms since body-conforming meshes are adopted here. The spatial discretization of the NS equations leads to a set of ordinary differential equations in real time:

$$(6) \quad \frac{d}{dt} (\mathbf{W}_{ijk} V_{ijk}) = -\mathbf{R}_{ijk} (\mathbf{W}).$$

where \mathbf{W} and \mathbf{R} are the vectors of cell conserved variables and residuals respectively. The convective terms are discretized using Osher's upwind scheme for its robustness, accuracy, and stability properties. MUSCL variable extrapolation is used to provide second-order accuracy with the Van Albada limiter to prevent spurious oscillations around shock waves. Boundary conditions are

set using ghost cells on the exterior of the computational domain. At the far-field, ghost cells are set at the freestream conditions. At solid boundaries the no-slip condition is set for viscous flows, or ghost values are extrapolated from the interior (ensuring the normal component of the velocity on the solid wall is zero) for Euler flow.

The integration in time of Eq. 6 to a steady-state solution is performed using a fully implicit time-marching scheme by:

$$(7) \quad \frac{\mathbf{W}_{ijk}^{n+1} - \mathbf{W}_{ijk}^n}{\Delta t} = -\frac{1}{V_{ijk}} \mathbf{R}_{ijk}(\mathbf{W}_{ijk}^{n+1}),$$

where $n + 1$ denotes the real time $(n + 1)\Delta t$. For steady state problems, the real time is replaced by a pseudo time (τ), that is also used for unsteady problems in the dual time stepping scheme of Jameson [45]. Equation 7 represents a system of non-linear algebraic equations and to simplify the solution procedure, the flux residual $\mathbf{R}_{ijk}(\mathbf{W}_{ijk}^{n+1})$ is linearized in time as follows:

$$(8) \quad \mathbf{R}_{ijk}(\mathbf{W}^{n+1}) \approx \mathbf{R}_{ijk}^n(\mathbf{W}^n) + \frac{\partial \mathbf{R}_{ijk}}{\partial \mathbf{W}_{ijk}} \Delta \mathbf{W}_{ijk},$$

where $\Delta \mathbf{W}_{ijk} = \mathbf{W}_{ijk}^{n+1} - \mathbf{W}_{ijk}^n$. Equation 7 now becomes the following linear system:

$$(9) \quad \left[\frac{V_{ijk}}{\Delta t} \mathbf{I} + \frac{\partial \mathbf{R}_{ijk}}{\partial \mathbf{W}_{ijk}} \right] \Delta \mathbf{W}_{ijk} = -\mathbf{R}_{ijk}^n(\mathbf{W}^n).$$

The left hand side of Eq. (9) is then rewritten in terms of primitive variables \mathbf{P} :

$$(10) \quad \left[\left(\frac{V_{ijk}}{\Delta t} \right) \frac{\partial \mathbf{W}_{ijk}}{\partial \mathbf{P}_{ijk}} + \frac{\partial \mathbf{R}_{ijk}}{\partial \mathbf{P}_{ijk}} \right] \Delta \mathbf{P}_{ijk} = -\mathbf{R}_{ijk}^n(\mathbf{W}^n),$$

and the resulting linear system is solved with a GCG (Generalized Conjugate Gradient) iterative solver [46]. Since at steady state the left hand side of Eq. (10) must go to zero, the Jacobian $\partial \mathbf{R} / \partial \mathbf{P}$ can be approximated by evaluating the derivatives of the residuals with a first-order scheme for the inviscid fluxes. The first-order Jacobian requires less storage and, being more dissipative, ensures a better convergence rate to the GCG iterations.

The steady state solver for the turbulent case is formulated and solved in an identical manner to that described previously for the mean flow. The eddy-viscosity is calculated from the latest values of k and ω (for example) and is used to advance both the mean flow solution and the turbulence solution. An approximate Jacobian is used for the source term by only taking into account the contribution of the dissipation terms \hat{D}_k and \hat{D}_ω , i.e. no account of the production terms is taken on the left hand side of the system.

The solver HMB2 also implements the harmonic balance method [43, 44], that allows for a direct calculation of a periodic state. The flow is assumed to be periodic with frequency ω and it is represented with N_H Fourier modes. The solution is split into $N_T = 2N_H + 1$ discrete equally spaced sub-intervals over the period $T = 2\pi/\omega$

$$(11) \quad \mathbf{W}_{\text{hb}} = \begin{pmatrix} \mathbf{W}(t_0 + \Delta t) \\ \mathbf{W}(t_0 + 2\Delta t) \\ \vdots \\ \mathbf{W}(t_0 + T) \end{pmatrix}, \quad \mathbf{R}_{\text{hb}} = \begin{pmatrix} \mathbf{R}(t_0 + \Delta t) \\ \mathbf{R}(t_0 + 2\Delta t) \\ \vdots \\ \mathbf{R}(t_0 + T) \end{pmatrix},$$

where $\Delta t = 2\pi/(N_T\omega)$. The harmonic balance equation then reads

$$(12) \quad \omega \mathbf{D} \mathbf{W}_{\text{hb}} + \mathbf{R}_{\text{hb}} = 0,$$

where

$$(13) \quad D_{ij} = \frac{2}{N_T} \sum_{k=1}^{N_H} k \sin(2\pi k(j-i)/N_T)$$

is the Fourier collocation derivative operator. Equation (12) is solved using a dual-time fully implicit method, a step of which is written as

$$(14) \quad \frac{\mathbf{W}_{\text{hb}}^{n+1} - \mathbf{W}_{\text{hb}}^n}{\Delta t} = -[\omega \mathbf{D} \mathbf{W}_{\text{hb}}^n + \mathbf{R}_{\text{hb}}(\mathbf{W}_{\text{hb}}^{n+1})].$$

5 FULLY IMPLICIT TANGENT AND ADJOINT SOLVERS

To compute aerodynamic sensitivities we need to solve either the linear system (3) for the tangent formulation or the discrete adjoint equations (4), and then use the sensitivity equations (2) or (5), respectively. Despite the tangent mode formulation being slightly more efficient for aeromechanics applications, due to the limited number of input parameters, the adjoint formulation has also been implemented, in view of future applications of the sensitivity equation approach in shape optimization problems.

The following discussion refers to the steady solver, but the presented method can be extended to the harmonic balance solver with few modifications. Indeed, to form the harmonic balance equations residual the steady residual functions are used for each of the harmonic snapshots, then the Fourier collocation derivative operator is added. For this reason the addition of the adjoint methods implicitly cover the harmonic balance functionality of the solver. Results with the harmonic balance method are also presented in subsequent paragraphs.

The linear system (3) of the tangent formulation and the linear system (4) of the adjoint formulation become stiff as the dimension of the flow problem increases, and therefore a suitable preconditioner is required to stabilize the solution algorithm. Another way to tackle the stiffness problem is to reformulate the linear system as a fixed-point iteration problem [47, 48], where an approximation of the linear system matrix, with better convergence properties, is introduced as a preconditioner to advance the solution at each iteration. Written in terms of primitive variables, the fixed-point iterative schemes reads:

$$(15) \quad \hat{J} \Delta \mathbf{P}_x^{n+1} = -\frac{\partial \mathbf{R}}{\partial \mathbf{x}} - \mathbf{J} \mathbf{P}_x^n, \quad \text{Tangent form}$$

$$(16) \quad \hat{J}^T \Delta \boldsymbol{\lambda}^{n+1} = -\left(\frac{\partial \mathbf{I}}{\partial \mathbf{P}}\right)^T - \mathbf{J}^T \boldsymbol{\lambda}^n, \quad \text{Adjoint form}$$

where

$$\begin{aligned} \mathbf{J} &= \frac{\partial \mathbf{R}}{\partial \mathbf{P}}, \quad \hat{J} = \left(\frac{V}{\Delta t}\right) \frac{\partial \mathbf{W}}{\partial \mathbf{P}} + \left[\frac{\partial \mathbf{R}}{\partial \mathbf{P}}\right]^{\text{1st}}, \\ \mathbf{P}_x &= \frac{\partial \mathbf{P}}{\partial \mathbf{x}}, \quad \Delta \mathbf{P}_x^{n+1} = \mathbf{P}_x^{n+1} - \mathbf{P}_x^n, \quad \text{and} \\ \Delta \boldsymbol{\lambda}^{n+1} &= \boldsymbol{\lambda}^{n+1} - \boldsymbol{\lambda}^n. \end{aligned}$$

The matrix \mathbf{J} represents the exact flow residual Jacobian. The natural choice for the preconditioner \hat{J} is the matrix used for the base flow iterative scheme (10), the sum of a stabilising time derivative term and of the first-order residual Jacobian, which approximates \mathbf{J} and is more diagonally dominant. The fixed point iteration (15) is solved using the GCG iterative solver. In the adjoint iteration (16) the system matrix is the transpose of the preconditioner \hat{J} , and the linear system can be solved with a slightly modified version of the GCG solver, that implicitly performs the matrix transposition.

The iteration schemes (15) and (16) do not require the full exact Jacobian \mathbf{J} , but only the matrix-vector product $\mathbf{J}\mathbf{v}$ or $\mathbf{J}^T\mathbf{v}$. As explained later in this section, the computer code to perform the former product can be obtained by AD in tangent mode of the flow steady residual function, whereas the code for the latter product can be obtained with AD of the same function in adjoint mode. This avoids storing \mathbf{J} , and hence the computation of sensitivities adds only a small memory overhead to the base solver.

There are two options for applying the preconditioning matrix \hat{J} . The first is to form explicitly \hat{J} by computing analytically the low-order Jacobian terms. This is done by the same code used for the base flow implicit method. Since the GCG solvers only requires the matrix-vector product, the second option is to use the automatically differentiated residual function to compute directly $\hat{J}\mathbf{v}$ for scheme (15) or $\hat{J}^T\mathbf{v}$ for scheme (16). This is accomplished by the same set of functions used for $\mathbf{J}\mathbf{v}$ and $\mathbf{J}^T\mathbf{v}$, with the only difference that the (differentiated) MUSCL extrapolation is not being performed. In this case, a special version of the GCG linear solver is invoked, that calls the differentiated code instead of the explicit matrix-vector product.

5.1 Computation of the product $\mathbf{J}\mathbf{v}$

To produce the matrix-vector product of the residual Jacobian \mathbf{J} and a generic vector \mathbf{v} we have isolated the CFD solver code that computes the steady flow residuals. In particular, the steps involved in the computation of the residuals have been grouped in the single function `steady_residual`, described by the pseudocode of figure 1 (the calls to the turbulence modelling functions have been omitted in this discussion for simplicity). The inputs for the function are the vector $\hat{\mathbf{X}}$ of mesh velocities, the vector \mathbf{N} of surface normals (the mesh metrics), the solution vector \mathbf{P} in primitive variables and the freestream Mach number M_∞ . The function produces as output the steady residual vector \mathbf{R} .

The differentiated version of `steady_residual` in tangent mode, named `steady_residual_d`, has been hand-coded, and it simply calls the differentiated version of the inner functions present in the original statements, as shown by the pseudo code in figure 2. The inner functions have been instead differentiated individually by means of the source transformation tool TAPENADE, operated in tangent mode. As a convention, the functions differentiated in tangent mode are identified by the postfix “_d” appended to the base name.

The application of the source transformation tool to the complex HMB2 solver functions, which are written in C language, was not straightforward, and some hand coding was needed before and after the AD. For each function to be differentiated, the required manual modifications were as follows.

```

steady_residual(Xdot, N, P, M, R)
{
    // Set the boundary and halo cells
    call set_boundary(Xdot, N, P, M);

    // Exchange data at block/inter-processor boundaries
    call exchange_halo_cells(P);

    // Calculate residual looping over the blocks
    do for each mesh block
    {
        // Compute inviscid terms with Osher's scheme
        call inviscid_Osher(Xdot, N, P, M, R);

        // Compute viscous terms
        call viscous(N, P, M, R);
    }
}

```

Figure 1: Pseudocode for the computation of the steady residual vector.

- 1) Before differentiation: The function was modified so that all input and output quantities appeared explicitly as dummy arguments in the function interface, eliminating any access through global variables.
 - Before differentiation: The “\$AD II-LOOP” directive was added before loops where each iteration does not depend on a value that is computed by another iteration. There are many loops of this kind in a typical CFD code, such as loops over the mesh elements to compute metric terms, fluxes and residuals, and loops over the surface mesh elements to perform loads integration. The directive instructs TAPENADE to treat this type of loops in a more efficient way when reverse differentiation is performed.
- 2) After differentiation: For efficiency, most of the data arrays in HMB2 are allocated in a contiguous memory area and accessed through pointer arithmetics. The differentiation tool sometimes does not handle correctly the pointer arithmetics and data access must be manually corrected in the generated code.

Note that, differently from the *ADjoint* method proposed by Mader and Martins [2], the present approach doesn't need rewriting of the residual evaluation code, transforming the original flux calculation loop over the computational control volume faces to a complete single control volume evaluation. This, in fact, is only needed to exploit the sparsity pattern of the Jacobian matrix when computing explicitly its elements. Since the present solver relies only on the computation of a single matrix-vector product $J\mathbf{v}$ or $J^T\mathbf{v}$ at each fixed-point iteration, the original CFD code loops can be differentiated directly both in forward and reverse mode, provided that the directive “\$AD II-LOOP” is specified before the loops to obtain an efficient code also when using the reverse mode.

The differentiated residual function `steady_residual_d` has the additional arguments $\delta\dot{\mathbf{X}}$, $\delta\mathbf{N}$, $\delta\mathbf{P}$, δM_∞ and $\delta\mathbf{R}$ (`Xdot_d`, `N_d`, `P_d`, `M_d` and `R_d` in the pseudocode, respectively) that represent the differentials of the quantities involved in the residuals computation. For any value of the input differentials, the action of `steady_residual_d` is to compute the consequent variation of the residual vector, that is,

$$(17) \quad \delta\mathbf{R} = \frac{\partial\mathbf{R}}{\partial\dot{\mathbf{X}}} \delta\dot{\mathbf{X}} + \frac{\partial\mathbf{R}}{\partial\mathbf{N}} \delta\mathbf{N} + \frac{\partial\mathbf{R}}{\partial\mathbf{P}} \delta\mathbf{P} + \frac{\partial\mathbf{R}}{\partial M_\infty} \delta M_\infty.$$

The third term in the right hand side is the product between the exact residuals Jacobian matrix with an arbitrary vector of solution variations. Thus, using `steady_residual_d` with $\delta\dot{\mathbf{X}} = 0$, $\delta\mathbf{N} = 0$, $\delta\mathbf{P} = \mathbf{P}_x^n$, $\delta M_\infty = 0$ produces the matrix-vector product necessary to compute the right hand side of the fixed-point iteration (15).

Note that additional memory is necessary to solve equation (3) via the fixed-point iterations (15) since storing the differentials $\delta\dot{\mathbf{X}}$, $\delta\mathbf{N}$, $\delta\mathbf{P}$ and $\delta\mathbf{R}$ is now needed. This, however, represents only 10-15% of the memory used by the implicit solver for the base flow.

5.2 Computation of the product $J^T\mathbf{v}$

The computation of the matrix-vector product $J^T\mathbf{v}$ requires the differentiation, in adjoint mode, of the steady residual function. As for the tangent mode case, the adjoint code for the main function `steady_residual_b` has been manually coded, while the inner functions have been differentiated using TAPENADE in reverse mode. The functions differentiated in adjoint mode are labelled by the addition of the postfix “_b” to the base name. The pseudocode for `steady_residual_b` is shown in figure 3.

Note that the adjoint code is more complex with respect to the tangent mode code. There are calls to the non-differentiated functions at the beginning, in what is called the *forward sweep*, where all the quantities needed during the subsequent back-propagation of the derivatives are calculated. The back-propagation of derivatives is performed in the *reverse sweep*, where the


```

steady_residual_d(Xdot, Xdot_d, N, N_d, P, P_d, M, M_d, R, R_d)
{
    // Set the boundary and halo cells
    call set_boundary_d(Xdot, Xdot_d, N, N_d, P, P_d, M, M_d);

    // Exchange data at block/inter-processor boundaries
    call exchange_halo_cells_d(P, P_d);

    // Calculate residual differentials looping over the blocks
    do for each mesh block
    {
        // Compute inviscid terms differentials
        call inviscid_Osher_d(Xdot, Xdot_d, N, N_d, P, P_d, M, M_d, R, R_d);

        // Compute viscous terms differentials
        call viscous_d(N, N_d, P, P_d, M, M_d, R, R_d);
    }
}

```

Figure 2: Pseudocode for the steady residual function differentiated in tangent mode.

differentiated versions of the functions called in the original statements are executed in reverse order. Also, not every call to non-differentiated function is present in the forward sweep of the adjoint code, like the call to `inviscid_osher` for instance, since the values computed by these functions are not needed during the reverse sweep.

For any value of the input residual differentials $\delta \mathbf{R}$ (`R_b` in the pseudocode), the action of the adjoint code in `steady_residual_b` is to compute the vectors $\delta \dot{\mathbf{X}}$, $\delta \mathbf{N}$, $\delta \mathbf{P}$ and δM_∞ (`Xdot_b`, `N_b`, `P_b` and `M_b` in the pseudocode, respectively) of weighted partial derivatives of the residuals:

$$(18) \quad \delta \dot{\mathbf{X}} = \left(\frac{\partial \mathbf{R}}{\partial \dot{\mathbf{X}}} \right)^T \delta \mathbf{R},$$

$$(19) \quad \delta \mathbf{N} = \left(\frac{\partial \mathbf{R}}{\partial \mathbf{N}} \right)^T \delta \mathbf{R},$$

$$(20) \quad \delta \mathbf{P} = \left(\frac{\partial \mathbf{R}}{\partial \mathbf{P}} \right)^T \delta \mathbf{R},$$

$$(21) \quad \delta M_\infty = \left(\frac{\partial \mathbf{R}}{\partial M_\infty} \right)^T \delta \mathbf{R}.$$

It is interesting to observe that the role of the dual variables $\delta \dot{\mathbf{X}}$, $\delta \mathbf{N}$, $\delta \mathbf{P}$, δM_∞ and $\delta \mathbf{R}$ in the differentiated function `steady_residual_d` is reversed in function `steady_residual_b`: input quantities in the former are output in the latter, and *vice versa*.

The vector $\delta \mathbf{P}$ is the product between the transpose of the exact residual Jacobian matrix and an arbitrary vector. It follows that a call of `steady_residual_b` with $\delta \mathbf{R} = \boldsymbol{\lambda}^n$ produces the matrix-vector product appearing in the right hand side of the fixed-point iteration (16).

The additional memory for `steady_residual_b` is due to the storage needed for the variables $\delta \dot{\mathbf{X}}$, $\delta \mathbf{N}$, $\delta \mathbf{P}$ and $\delta \mathbf{R}$. This needs to be added to the memory allocated temporarily by the inner functions differentiated in reverse mode by TAPENADE. The reverse mode differentiated code requires, in fact, to save some of the quantities computed during the forward sweep, when they are necessary to back-propagate derivatives at certain stages of the reverse sweep. The temporary storage allocated by the reverse differentiated routines called by `steady_residual_b` is, however, very small.

6 AERODYNAMIC SENSITIVITIES FOR FIXED WING CASES

Even if the main objective of the paper is the computation of aerodynamic derivatives for rotorcraft, the implementation of the discrete adjoint for airfoils and fixed wing cases is a necessary intermediate step to test the method. Moreover, this allows for the comparison of sensitivity results obtained with HMB2 with established cases in the literature.

In the previous section we described how to solve the linear system yielding the derivative $\partial \mathbf{P} / \partial x$ for the tangent method or the adjoint variables vector $\boldsymbol{\lambda}$ for the adjoint method. Here we briefly describe how to compute the other terms of the sensitivity equations (2) and (5), namely $\partial \mathbf{R} / \partial x$, $\partial I / \partial \mathbf{P}$ and $\partial I / \partial x$. For fixed wing aircraft, the outputs I of interest shall be any of the force and moment coefficients C_L , C_D , C_Y , C_l , C_m and C_n , while the independent parameters x shall be either the incidence α , the sideslip β , the freestream Mach number M_∞ or any of the three rotational rates p , q and r around the wind axes (see figure 4).

```

steady_residual_b(Xdot, Xdot_b, N, N_b, P, P_b, M, M_b, R, R_b)
{
    // Set the boundary and halo cells
    call set_boundary(Xdot, N, P, M);

    // Exchange data at block/inter-processor boundaries
    call exchange_halo_cells(P);

    // Calculate residual differentials looping over the blocks
    do for each mesh block
    {
        // Compute viscous terms differentials
        call viscous_b(N, N_b, P, P_b, M, M_b, R, R_b);

        // Compute inviscid terms differentials
        call inviscid_Osher_b(Xdot, Xdot_b, N, N_b, P, P_b, M, M_b, R, R_b);
    }

    // Exchange differentials at block/inter-processor boundaries
    call exchange_halo_cells_b(P_b);

    // Set the differentials at boundary and halo cells
    call set_boundary_b(Xdot, Xdot_b, N, N_b, P, P_b, M, M_b);
}

```

Figure 3: Pseudocode for the steady residual function differentiated in adjoint mode.

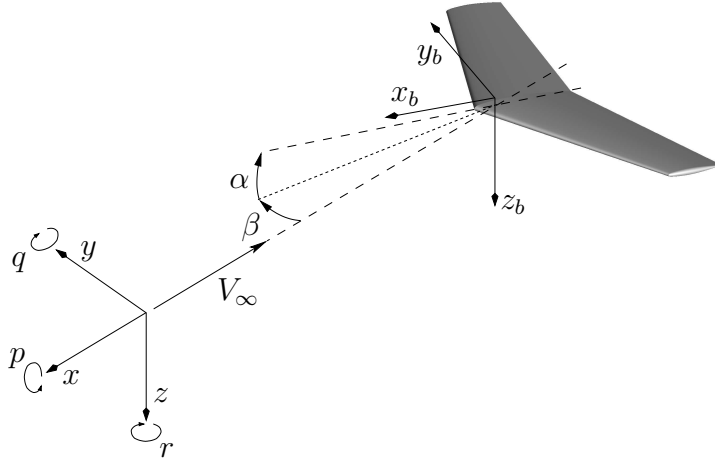


Figure 4: Flight-mechanic variables for the fixed wing.

6.1 Treatment of the terms $\partial \mathbf{R} / \partial \alpha$, $\partial \mathbf{R} / \partial \beta$

These partial derivatives represent the variation of the residual vector \mathbf{R} due to a variation of the angle of attack or sideslip. For the computation of any of these two terms it is convenient to express the derivative as follows:

$$(22) \quad \frac{\partial \mathbf{R}}{\partial x} = \frac{\partial \mathbf{R}}{\partial \mathbf{N}} \frac{\partial \mathbf{N}}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial x}, \quad x \in \{\alpha, \beta\},$$

where the first derivative in the right hand side is the variation of the residual due to a change in the mesh metrics (vector \mathbf{N}), the second is the variation of the metrics due to a change in mesh coordinates (vector \mathbf{X}), and the third is the variation of the mesh coordinates given by a change in the input parameter.

Recalling that the action of the steady residual function differentiated in tangent mode is given by Eq. (17), the desired term $\partial \mathbf{R} / \partial x$ can be computed by a single call to `steady_residual_d` with the following input arguments set as follows: $\delta \dot{\mathbf{X}} = 0$, $\delta \mathbf{N} = (\partial \mathbf{N} / \partial \mathbf{X}) \cdot (\partial \mathbf{X} / \partial x)$, $\delta \mathbf{P} = 0$, $\delta M_\infty = 0$. The term $\partial \mathbf{N} / \partial \mathbf{X}$ can be obtained by tangent differentiation of the function computing the mesh metrics, while the term $\partial \mathbf{X} / \partial x$, with $x \in \{\alpha, \beta\}$, can be computed directly, since it represents the variation of the mesh coordinates due to a variation of angle of attack or sideslip, respectively.

The chain rule expansion (22) is suitable to introduce the angle of attack or the sideslip variation as a rigid rotation or a deformation of the computational mesh, depending on how the rightmost term $\partial \mathbf{X} / \partial x$ is computed. Both methods are implemented in HMB2. The rigid rotation technique is faster and easier to implement. On the other hand, the grid deformation technique is more

general and can be used also to compute the sensitivity with respect to arbitrary shape change of the airfoil or fixed wing aircraft. Also, the grid deformation method is necessary to introduce variations of the blade control angles in rotorcraft applications, where the grid cannot be simply rotated, since that would alter the rotation axis of the rotor.

Another method for computing the angle of attack and sideslip sensitivities is to alter the far-field boundary conditions, so as to introduce the required variation of the freestream velocity direction. In this case, the residual derivative is expressed as

$$(23) \quad \frac{\partial \mathbf{R}}{\partial x} = \frac{\partial \mathbf{R}}{\partial \mathbf{W}_{\text{ff}}} \frac{\partial \mathbf{W}_{\text{ff}}}{\partial x}, \quad x \in \{\alpha, \beta\},$$

where \mathbf{W}_{ff} is the flow solution restricted to the far-field boundary. This approach is also easy to implement, since the evaluation of $\partial \mathbf{W}_{\text{ff}} / \partial x$ is straightforward. Results based on both the chain rule expansions (22) and (23) will be given in the paragraphs dedicated to numerical results.

6.2 Treatment of the terms $\partial \mathbf{R} / \partial p$, $\partial \mathbf{R} / \partial q$, $\partial \mathbf{R} / \partial r$

. These partial derivatives represent the variation of the residual vector \mathbf{R} due to a variation of the rotational speeds around the three wind axes. Thanks to the arbitrary Lagrangian–Eulerian formulation of the flow equations in HMB2, these terms can be obtained through the following chain rule expansion:

$$(24) \quad \frac{\partial \mathbf{R}}{\partial x} = \frac{\partial \mathbf{R}}{\partial \dot{\mathbf{X}}} \frac{\partial \dot{\mathbf{X}}}{\partial x}, \quad x \in \{p, q, r\},$$

where the first derivative in the right hand side is the variation of the residual due to a change in the mesh velocities, and the second derivative is the variation of the mesh velocities given by a change in the input parameter. The desired term $\partial \mathbf{R} / \partial x$ can be computed by a single call to `steady_residual_d` with $\delta \dot{\mathbf{X}} = \partial \dot{\mathbf{X}} / \partial x$, $\delta \mathbf{N} = 0$, $\delta \mathbf{P} = 0$, $\delta M_{\infty} = 0$. The term $\partial \dot{\mathbf{X}} / \partial x$, with $x \in \{p, q, r\}$, is relatively easy to compute, as it represents the variation of the mesh velocities due to a change in the rotational speed around the three wind axis.

6.3 Treatment of the term $\partial \mathbf{R} / \partial M_{\infty}$

This partial derivative represents the dependence of the residual vector upon the freestream Mach number. Since the Mach number appears explicitly in the HMB2 formulation, and hence in the steady residual code, this term can be computed by calling `steady_residual_d` with $\delta \dot{\mathbf{X}} = 0$, $\delta \mathbf{N} = 0$, $\delta \mathbf{P} = 0$, $\delta M_{\infty} = 1$.

6.4 Treatment of the terms $\partial I / \partial \alpha$, $\partial I / \partial \beta$

These partial derivatives represent the direct dependence of force and moment coefficients upon the variation of angle of attack or sideslip. They can be computed in a similar way to the residual terms above, by first expressing the derivative as

$$(25) \quad \frac{\partial I}{\partial x} = \frac{\partial I}{\partial \mathbf{N}} \frac{\partial \mathbf{N}}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial x}, \quad x \in \{\alpha, \beta\}.$$

The term $\partial I / \partial \mathbf{N}$ expresses the dependence of force and moments upon the mesh metrics and is obtained by differentiating the function for computing the integrated loads in tangent mode with respect to mesh metrics. The other terms have been already discussed above.

6.5 Treatment of the terms $\partial I / \partial p$, $\partial I / \partial q$, $\partial I / \partial r$

These partial derivatives are zero, because there is no direct dependence of force and moment coefficients upon the rotational speeds around the wind axes.

6.6 Treatment of the term $\partial I / \partial M_{\infty}$

This partial derivative represents the dependence of the force coefficient upon the freestream Mach number. It is obtained by differentiating the function for computing the integrated loads in tangent mode with respect to the Mach number.

6.7 Treatment of the term $\partial I / \partial \mathbf{P}$

These partial derivatives represent the variation of force and moment coefficients due to a variation of the flow variables. They can be efficiently computed by differentiating the function for computing the integrated loads with respect to the flow variables in adjoint mode. The use of the adjoint mode for this computation is justified by the fact that the input is represented by all the flow variables, which clearly outnumber the outputs, represented by the six force and moment coefficients.

7 AERODYNAMIC SENSITIVITIES FOR ROTORS IN HOVER AND FORWARD FLIGHT

Hover computations are performed in HMB2 formulating the equations in the rotating reference frame of the rotor, so that the solution is steady. Periodic boundary conditions are also used to take advantage of the symmetry of the problem, allowing for the discretization of a single rotor blade.

With respect to the sensitivities of rotorcraft in hover, the outputs I of interest are the thrust coefficient $C_T = T/(\rho A \Omega^2 R^2)$ and the torque coefficient $C_Q = Q/(\rho A \Omega^2 R^3)$, while the independent parameters x are the collective pitch θ , the flap angle β and the vertical velocity w (see figure 5). A brief description of the sensitivity equation terms required to compute the aerodynamic derivatives for rotors in hovering flight follows.

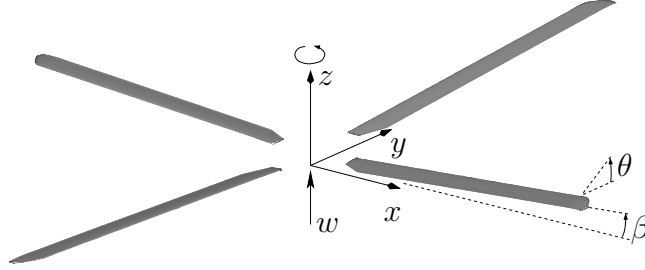


Figure 5: Flight-mechanic variables for the rotor in hover.

7.1 Treatment of the terms $\partial \mathbf{R}/\partial \theta$, $\partial \mathbf{R}/\partial \beta$

These partial derivatives represent the variation of the residual vector \mathbf{R} due to a change of the pitch or flap angle. For the computation of any of these two terms it is convenient to express the derivative as follows:

$$(26) \quad \frac{\partial \mathbf{R}}{\partial x} = \frac{\partial \mathbf{R}}{\partial \mathbf{N}} \frac{\partial \mathbf{N}}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial x}, \quad x \in \{\theta, \beta\}.$$

The first two derivatives in the right hand side can be computed with the same method used for the fixed wing aircraft case. The third derivative, $\partial \mathbf{X}/\partial x$, with $x \in \{\theta, \beta\}$, is computed by differentiation in tangent mode of a mesh deformation function based on Inverse Distance Weighting (IDW) [49]. This function evaluates the mesh differentials due to a rigid rotation around the pitch or flap axis of the blade surface, and of the consequent volume mesh deformation, which is set up so as to keep the other mesh boundaries (periodic planes and far-field) fixed.

7.2 Treatment of the term $\partial \mathbf{R}/\partial w$

This partial derivatives represent the variation of the residual vector \mathbf{R} due to a variation of the velocity along the rotor axis of rotation. The computation of these terms requires to express the derivative as:

$$(27) \quad \frac{\partial \mathbf{R}}{\partial w} = \frac{\partial \mathbf{R}}{\partial \dot{\mathbf{X}}} \frac{\partial \dot{\mathbf{X}}}{\partial w}.$$

The first derivative has been already discussed. The second derivative is easy to compute, as it represents a uniform variation of the mesh velocities in the direction of the rotor rotation axis.

7.3 Treatment of the terms $\partial I/\partial \theta$, $\partial I/\partial \beta$

These partial derivatives represents the direct dependence of force and moment coefficients upon the pitch and flap angle. They can be computed in a similar way to the $\partial I/\partial \alpha$ and $\partial I/\partial \beta$ terms for the fixed wing aircraft, by expressing the derivative as

$$(28) \quad \frac{\partial I}{\partial x} = \frac{\partial I}{\partial \mathbf{N}} \frac{\partial \mathbf{N}}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial x}, \quad x \in \{\alpha, \beta\},$$

and computing $\partial \mathbf{X}/\partial x$ as described here above for the derivatives of the residual vector with respect to the same input variables.

The computation of the flight-mechanic derivatives of rotors in forward flight can be re-casted into a steady problem by using the harmonic balance method [44]. All the above considerations about the partial derivatives of quantities related to the rotor in

hover apply to the forward flight case as well. In this last case, any perturbation to the independent parameters is introduced individually into each time snapshot of the solution. The corresponding variation of the outputs can be, for instance, averaged over the time snapshots to produce derivatives averaged over the rotor revolution.

8 NUMERICAL RESULTS

8.1 NACA0012 Aerofoil in Inviscid Flow

The first test case considers the inviscid flow around a NACA0012 airfoil. Sensitivity computations for this airfoil have been performed by Limache and Cliff [1] and by Mader *et al.* [2]. A mesh-convergence study was first conducted to select a reference mesh for subsequent computations. Table 2 shows the values of the flight-mechanic derivatives for the airfoil at zero incidence and $M_\infty = 0.5$, obtained solving the sensitivity equation in tangent mode on meshes of increasing density. The pseudotime Courant–Friedrichs–Lewy (CFL) for all the computations was set to 40. Figure 6 shows, for instance, the convergence of the derivative $\partial C_L / \partial \alpha$. As expected, the convergence is second order in space, and the fine mesh with 134,160 cells has been selected for comparison with published results. The comparison between HMB2 in tangent mode, HMB2 in adjoint mode and results taken from Limache and Cliff [1] and from Mader *et al.* [2] is given in table 3. The reference point for the moment coefficient in the table is assumed to be at the leading edge, and the same convention is used for all the airfoil test cases presented in the paper. As can be seen, the difference between the HMB2 sensitivities computed in tangent and adjoint mode is negligible, and the results are equal up to the eleventh significant digit, and thus proving the consistency of the method. The difference between the HMB2 and the reference results is below 2.2% which, considering the different meshes and discretization methods, is considered to be low.

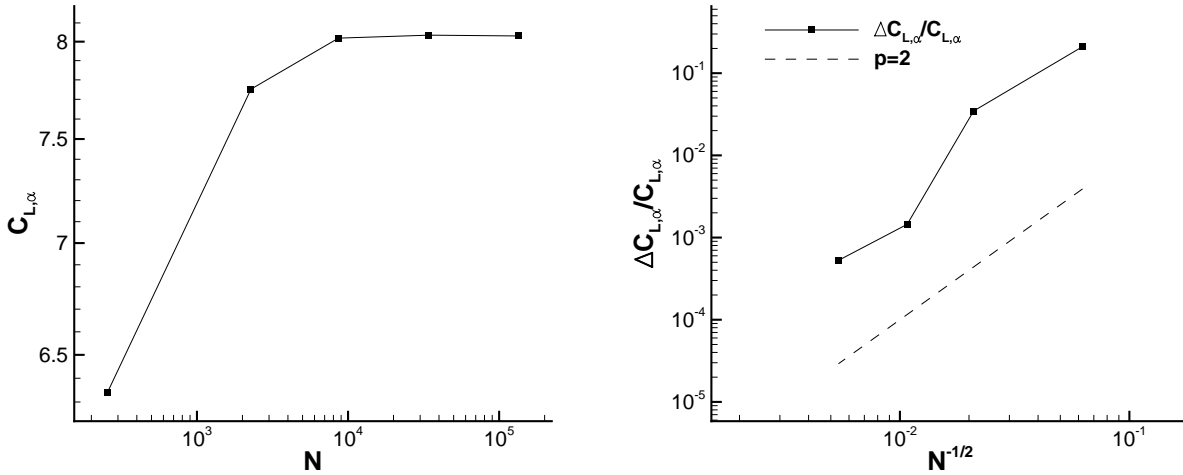


Figure 6: HMB2 mesh-convergence study for the NACA0012 airfoil (inviscid flow, $\alpha = 0^\circ$, $M_\infty = 0.5$).

Mesh size (# cells)	$\partial C_L / \partial \alpha$	$\partial C_M / \partial \alpha$	$\partial C_L / \partial q$	$\partial C_M / \partial q$
256	6.1521311	-1.5334439	8.8890361	-2.8771683
2278	7.6824618	-2.0060088	11.2133409	-3.7773141
8646	8.0042893	-2.1020533	11.6491348	-3.9194602
34320	8.0287614	-2.1102439	11.6730662	-3.9270310
134160	8.0271676	-2.1099737	11.6669031	-3.9250582

Table 2: HMB2 mesh convergence study for the NACA0012 airfoil (inviscid flow, $\alpha = 0^\circ$, $M_\infty = 0.5$).

The airfoil aerodynamic derivatives with respect to the angle of attack were computed using the chain rule expansion (22) and by rigidly rotating the mesh. As discussed in section 6, there are alternative methods to compute these derivatives. In fact, one might use the same expansion (22) to introduce the variation of the airfoil angle of attack with a mesh deformation. A third method uses the chain rule expansion (23) and alters the angle of attack by modifying the far-field boundary conditions for the flow velocity. The results obtained with the three methods are compared in table 4. The mesh rotation and the rotation of the far-field velocity gave close results, with a discrepancy below 0.1%. The grid deformation method gave derivatives which differ more, since it implies a deformation of the mesh, but the discrepancy with respect to the grid rotation method is still below 1%.

	$\partial C_L / \partial \alpha$	$\partial C_M / \partial \alpha$	$\partial C_L / \partial q$	$\partial C_M / \partial q$
HMB2 AD tangent	8.027167592289	-2.109973708883	11.666903146962	-3.925058240904
HMB2 AD adjoint	8.027167592297	-2.109973708886	11.666903146966	-3.925058240906
Limache and Cliff [1]			11.847000000000	-3.968000000000
Difference [%]			1.52	1.08
Mader <i>et al.</i> [2]	7.961756758205	-2.068623684859	11.921373826019	-3.999949642440
Difference [%]	0.82	2.00	2.13	1.87

Table 3: Comparison of HMB2 sensitivity results on the fine mesh with reference results for the NACA0012 airfoil (inviscid flow, $\alpha = 0^\circ$, $M_\infty = 0.5$).

Method	$\partial C_L / \partial \alpha$	$\partial C_M / \partial \alpha$
Grid rotation	8.0271674	-2.1099736
Far-field velocity rotation	8.0305361	-2.1115952
Grid deformation	7.9846367	-2.0889397

Table 4: Comparison of angle-of-attack sensitivity results for the NACA0012 airfoil obtained with different methods (inviscid flow, $\alpha = 0^\circ$, $M_\infty = 0.5$).

8.2 NACA0012 Aerofoil in Laminar Flow

We consider the viscous laminar flow around a NACA0012 airfoil at the same flight conditions used for the inviscid results ($\alpha = 0^\circ$, $M_\infty = 0.5$). The Reynolds number is 2×10^3 , and is sufficiently low to guarantee laminar flow. The computational mesh has 134,160 cells, and the first layer above the airfoil surface is at $10^{-3}c$.

The flow is laminar and a thick boundary layer develops around the airfoil. There are no sensitivity data available in the literature for the viscous case and therefore the results were compared with finite difference evaluation of the flow derivatives. The comparison between HMB2 in tangent mode, HMB2 in adjoint mode and second order centred finite differences is given in table 5. The difference between the HMB2 sensitivities computed in tangent and adjoint mode is negligible. The HMB2 and finite differences results are also in good agreement, the difference being below 0.1%. Note that the computed lift curve slope coefficient is in agreement with the values found in the literature for the NACA0012 airfoil in the low-Reynolds regime [50].

	$\partial C_L / \partial \alpha$	$\partial C_D / \partial \alpha$	$\partial C_M / \partial \alpha$
HMB2 AD tangent	3.0503815	≈ 0	-0.5653114
HMB2 AD adjoint	3.0503815	≈ 0	-0.5653114
HMB2 FD	3.0511247	≈ 0	-0.5655851
Difference [%]	0.024		0.048

Table 5: Comparison of angle-of-attack sensitivity computed with AD and FD for the NACA0012 airfoil ($\alpha = 0^\circ$, $M_\infty = 0.5$, $Re = 2000$).

8.3 NACA0012 Aerofoil in Turbulent Flow

An assessment of the computation of aerodynamic derivatives for turbulent flows has also been performed, based on a test case for the NACA0012 airfoil. The conditions were $\alpha = 0^\circ$, $M_\infty = 0.5$, $Re = 10^6$. The $k-\omega$ turbulence model [51] was used. The computational mesh had 53,206 cells, and the first cell layer above the airfoil surface was at $5 \cdot 10^{-6}c$.

The aerodynamic derivatives computed with the sensitivity equation were compared with finite difference results. The comparison between HMB2 in tangent mode, HMB2 in adjoint mode and second order centred finite differences is shown in table 6. The sensitivities computed in tangent and adjoint mode agree very well. As can be seen, the HMB2 and finite differences results are also in excellent agreement, confirming the validity of the method.

Another comparison has been attempted at a higher Mach number $M_\infty = 0.8$, where the turbulence model is more tightly coupled to the base flow, due to the interaction between the boundary layer and the shock waves. The aerodynamic derivatives for this test case are reported in table 7, which shows a very good agreement between AD and FD results. This is due to the exact account of all the terms coupling the turbulence model with the base flow, which have been differentiated with AD without introducing any approximations or simplifying hypotheses, such as frozen turbulence.

The derivation of the adjoint equations is particularly difficult when using the continuous adjoint approach, and often the simplifying approximation of frozen turbulence is introduced, which amounts in neglecting the derivatives of the turbulent variables.

	$\partial C_L / \partial \alpha$	$\partial C_D / \partial \alpha$	$\partial C_M / \partial \alpha$
HMB2 AD tangent	7.0912701	≈ 0	-1.7306609
HMB2 AD adjoint	7.0912701	≈ 0	-1.7306609
HMB2 FD	7.0923431	≈ 0	-1.7307926
Difference [%]	0.015		0.008

Table 6: Comparison of angle-of-attack sensitivity computed with AD and FD for the NACA0012 airfoil (k - ω turbulence model, $\alpha = 0^\circ$, $M_\infty = 0.5$, $\text{Re} = 10^6$).

	$\partial C_L / \partial \alpha$	$\partial C_D / \partial \alpha$	$\partial C_M / \partial \alpha$
HMB2 AD tangent	13.3220459	≈ 0	-4.2429812
HMB2 AD adjoint	13.3220459	≈ 0	-4.2429812
HMB2 FD	13.3179764	≈ 0	-4.2234059
Difference [%]	0.310		0.460

Table 7: Comparison of angle-of-attack sensitivity computed with AD and FD for the NACA0012 airfoil (k - ω turbulence model, $\alpha = 0^\circ$, $M_\infty = 0.8$, $\text{Re} = 10^6$).

It is interesting to quantify the inaccuracies introduced into the gradients by such approximation, as in Dwight and Brezillon [36] and in Lyu *et al.* [37] for the Spalart–Allmaras one-equation turbulence model, and in Marta *et al.* [38] for the k - ω two-equation model. In a discrete adjoint solver, the frozen turbulence can be simulated by simply disabling the differentiated code computing the turbulence model derivatives. Tables 8 and 9 compare the exact angle of attack derivatives with those computed with the frozen turbulence approximation for the NACA0012 subsonic and transonic cases, respectively. For the subsonic case the frozen turbulence assumption introduced a small error on the derivatives, which is below 1%. On the other hand, when the flow is transonic, the derivatives computed with the frozen turbulence are inaccurate, exhibiting a 28% error on the lift coefficient and 47% error on the moment coefficient. The pressure sensitivity with respect to the angle of attack for the transonic case is shown in figure 7, for both the exact and the frozen turbulence solutions. By comparing the two plots it follows that the frozen turbulence approximation leads to a significant underestimation of the solution variation in the shock region, which explains the large difference in the predicted aerodynamic derivatives.

	$\partial C_L / \partial \alpha$	$\partial C_D / \partial \alpha$	$\partial C_M / \partial \alpha$
Exact	7.0912701	≈ 0	-1.7306609
Frozen turbulence	7.1207804	≈ 0	-1.7354227
Difference [%]	0.414		0.274

Table 8: Comparison of angle-of-attack sensitivity computed with and without frozen turbulence approximation for the NACA0012 airfoil (k - ω turbulence model, $\alpha = 0^\circ$, $M_\infty = 0.5$, $\text{Re} = 10^6$).

	$\partial C_L / \partial \alpha$	$\partial C_D / \partial \alpha$	$\partial C_M / \partial \alpha$
Exact	13.3220459	≈ 0	-4.2429812
Frozen turbulence	10.3413015	≈ 0	-2.8930371
Difference [%]	27.841		46.662

Table 9: Comparison of angle-of-attack sensitivity computed with and without frozen turbulence approximation for the NACA0012 airfoil (k - ω turbulence model, $\alpha = 0^\circ$, $M_\infty = 0.8$, $\text{Re} = 10^6$).

8.4 NACA0012 Oscillating Aerofoil in Inviscid Flow

The differentiated code for the harmonic balance method has also been assessed with the CT1 test case taken from the AGARD database [52, 53], relative to an oscillating NACA0012 airfoil. The variation of the pitch angle is given by

$$(29) \quad \alpha(t) = \alpha_0 + \alpha_1 \sin(\omega t),$$

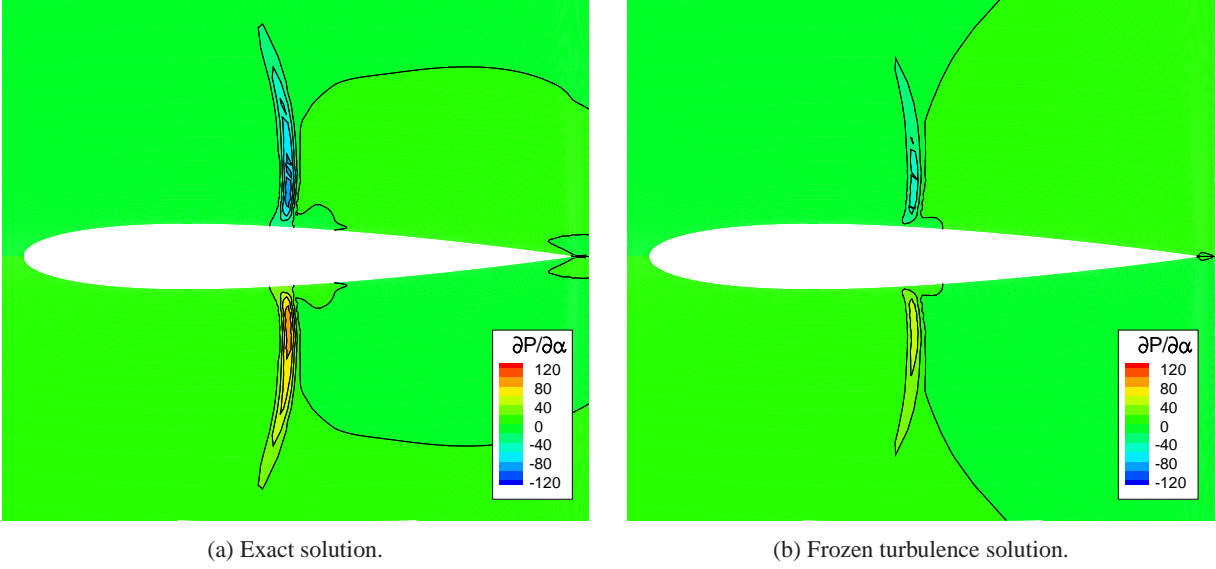


Figure 7: Pressure sensitivity to the angle of attack (NACA0012, k - ω turbulence model, $\alpha = 0^\circ$, $M_\infty = 0.8$, $Re = 10^6$).

where ω is related to the reduced frequency k by:

$$(30) \quad k = \frac{\omega c}{2U_\infty}.$$

The case freestream Mach number was 0.6, the mean incidence was $\alpha_0 = 2.89^\circ$, $\alpha_1 = 2.41^\circ$, $k = 0.0808$, and the pitching motion was about the quarter chord. The flow was inviscid and the computational mesh had 8,646 cells.

The cycle averaged derivatives of the coefficients C_L , C_D and C_M with respect to α_0 , M_∞ and q have been computed with HMB2 solving the sensitivity equation in tangent and in adjoint modes. These results are displayed in tables 10–12, and compared to those obtained with second order centred finite differences. The tangent and adjoint computations are consistent, and good agreement between AD and FD results is observed, with a relative error below 1% for all the derivatives.

	$\partial C_L / \partial \alpha_0$	$\partial C_D / \partial \alpha_0$	$\partial C_M / \partial \alpha_0$
HMB2 AD tangent	8.6444957	0.0167310	0.0957970
HMB2 AD adjoint	8.6444957	0.0167310	0.0957970
HMB2 FD	8.6447053	0.0172026	0.0960558
Difference [%]	0.002	0.404	0.270

Table 10: Comparison of angle-of-attack sensitivities computed with AD and FD for the oscillating NACA0012 airfoil (inviscid flow, $\alpha_0 = 2.89^\circ$, $\alpha_1 = 2.41^\circ$, $k = 0.0808$, $M_\infty = 0.6$).

	$\partial C_L / \partial M_\infty$	$\partial C_D / \partial M_\infty$	$\partial C_M / \partial M_\infty$
HMB2 AD tangent	0.4831079	0.0301123	0.0957970
HMB2 AD adjoint	0.4831079	0.0301123	0.0621395
HMB2 FD	0.4867285	0.0298607	0.0619160
Difference [%]	0.749	0.835	0.360

Table 11: Comparison of Mach number sensitivities computed with AD and FD for the oscillating NACA0012 airfoil (inviscid flow, $\alpha_0 = 2.89^\circ$, $\alpha_1 = 2.41^\circ$, $k = 0.0808$, $M_\infty = 0.6$).

8.5 NACA0012 Oscillating Aerofoil in Turbulent Flow

A more demanding test case is represented by the NACA0012 oscillating airfoil in turbulent flow, whose adjoint problem is particularly stiff and hard to solve. In fact, the Fourier collocation operator of the harmonic balance method adds extra-diagonal terms to the Jacobian matrix that reduce its diagonal dominance. Also, the coupling terms of the turbulence model further increase the Jacobian matrix stiffness.

	$\partial C_L / \partial q$	$\partial C_D / \partial q$	$\partial C_M / \partial q$
HMB2 AD tangent	12.351258	-0.1060475	-0.8792168
HMB2 AD adjoint	12.351258	-0.1060475	-0.8792168
HMB2 FD	12.361514	-0.1066636	-0.8801156
Difference [%]	0.083	0.581	0.102

Table 12: Comparison of pitching rate sensitivities computed with AD and FD for the oscillating NACA0012 airfoil (inviscid flow, $\alpha_0 = 2.89^\circ$, $\alpha_1 = 2.41^\circ$, $k = 0.0808$, $M_\infty = 0.6$).

The pitching motion of the airfoil is given by Eq. (29), with $\alpha_0 = 1^\circ$, $\alpha_1 = 1^\circ$ and $k = 0.03668$. The case freestream Mach number was $M_\infty = 0.55$ and the Reynolds number was $\text{Re} = 10^6$. The turbulent viscosity was accounted for using the Wilcox $k-\omega$ turbulence model [51]. The pseudotime CFL was set to 5 for both the base flow and the turbulence-model equations. The mesh is the same used for the steady case described in section 8.3.

The cycle averaged derivatives of the coefficients C_L , C_D and C_M for the pitching airfoil computed with AD in adjoint mode and with FD are shown in table 13. The fully implicit solver drives the adjoint equations relative residual below 10^{-10} , resulting in accurate derivatives. There is a good agreement between AD and FD derivatives, the relative error being lower than 1% for all the derivatives.

	$\partial C_L / \partial \alpha_0$	$\partial C_D / \partial \alpha_0$	$\partial C_M / \partial \alpha_0$	$\partial C_L / \partial q$	$\partial C_D / \partial q$	$\partial C_M / \partial q$
HMB2 AD adjoint	7.3918769	0.0149052	0.0542477	8.2267771	-0.0671146	-1.5290074
HMB2 FD	7.3976873	0.0149701	0.0538260	8.2136895	-0.0667408	-1.5283029
Difference [%]	0.079	0.435	0.777	0.159	0.557	0.046

Table 13: Comparison of angle-of-attack and pitching rate sensitivities computed with AD for the pitching NACA0012 airfoil ($k-\omega$ turbulence model, $\alpha_0 = 1^\circ$, $\alpha_1 = 1^\circ$, $k = 0.03668$, $M_\infty = 0.55$, $\text{Re} = 10^6$).

8.6 ONERA-The French Aerospace Lab M6 Wing in Inviscid Flow

A three-dimensional test case considers the inviscid flow around an ONERA-The French Aerospace Lab M6 wing in transonic flight, at $M_\infty = 0.8395$ and $\alpha = 3.06^\circ$. Again, the HMB2 results have been compared with the reference results of Mader *et al.* [2]. The HMB2 mesh is composed by 4 million cells (see figure 8), while the reference mesh [2] has 14.7 million cells. The pseudotime CFL for the base flow and for the tangent/adjoint solutions was set to 20.

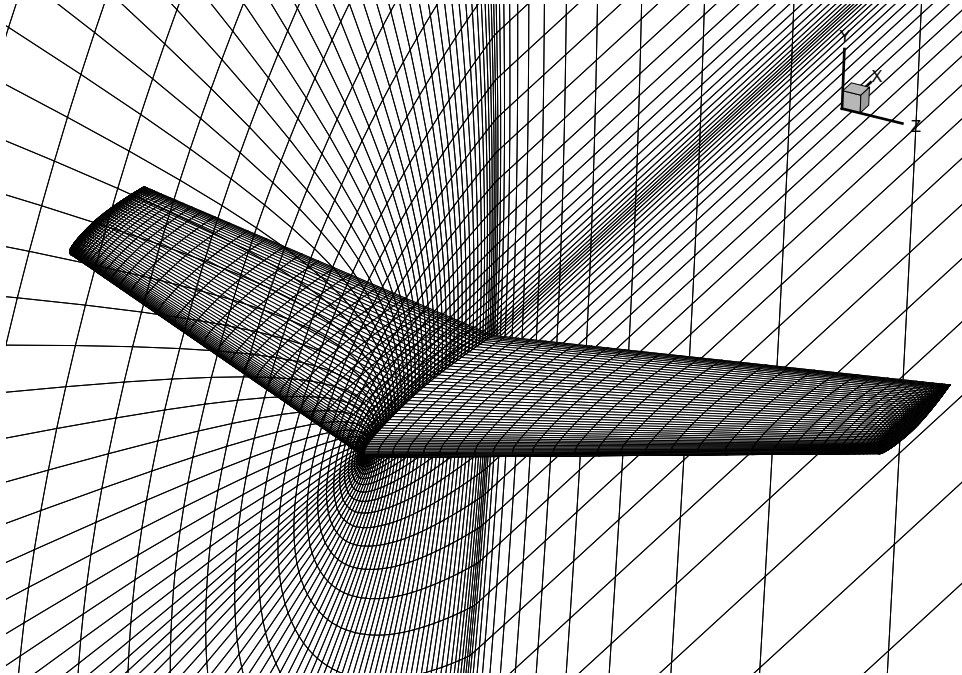


Figure 8: Mesh for the ONERA-The French Aerospace Lab M6 wing (every second mesh line is shown).

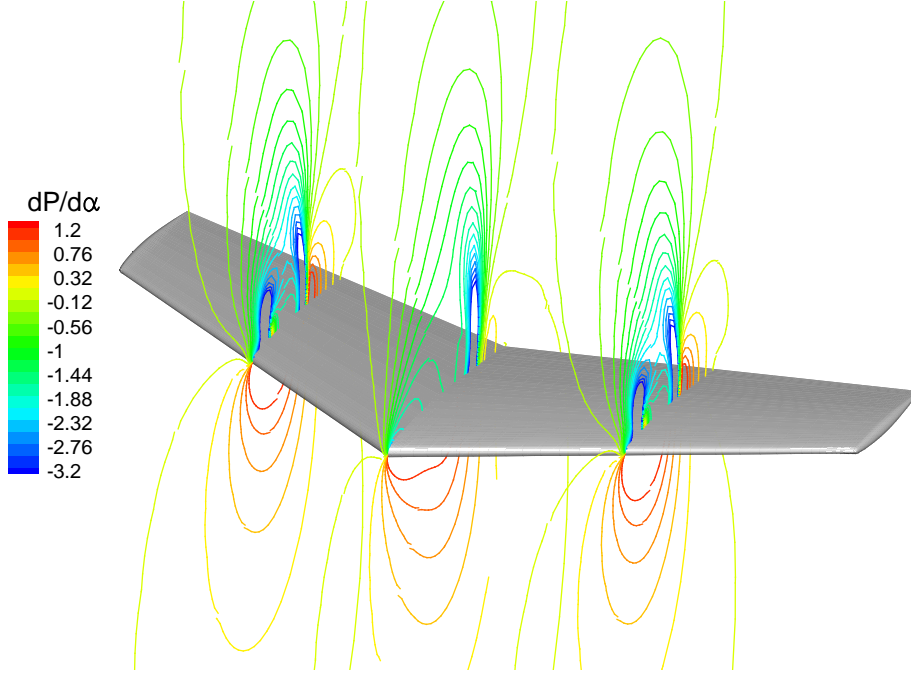


Figure 9: Pressure sensitivity to the angle of attack (M6, inviscid flow, $\alpha = 3.06^\circ$, $M_\infty = 0.8395$).

Figure 9 shows contours of the pressure sensitivity with respect to the angle of attack, extracted from the solution to equation (3). As expected, the highest values of the pressure derivative are located in the shock region, while the lower surface presents a more regular positive pressure variation. The comparison between HMB2 and reference flight-mechanic derivatives is reported in table 14. The HMB2 results are in good agreement with the reference results, with limited exceptions for the derivatives involving integration in the direction parallel to the wing planform and the derivatives with respect to the freestream Mach number. The discrepancies of the former derivatives are not surprising, since the forces parallel to wing planform are very small and the results are therefore significantly mesh dependent. Additional differences between the present and published results may be due to different numerical schemes and Riemann solvers employed.

Regarding the performance of the method, the time for computing the base flow with a relative convergence of the L_2 error norm of 10^{-9} was 1 hour on 32 cores (2.2GHz Intel Xeon E5-2660). For the tangent method, the computational time for a single sensitivity computation with a relative convergence of 10^{-9} was between 70 and 90% the time needed by the base flow solver, depending on the input variable. For the adjoint method, the computational time was between 80 and 100% of the base flow solver time, depending on the output quantity. The overall time for the aerodynamic sensitivities computation was 4.8 hours for the tangent mode solver and 5.5 hours for the adjoint solver. Note that for computing all the aerodynamic derivatives with second order centred finite differences, 12 CFD solutions would have been needed, for an overall computational time of about 12 hours.

8.7 Inviscid ONERA-The French Aerospace Lab 7AD Rotor in Hover

To verify the computation of aerodynamic sensitivities for rotors, the inviscid flow around the ONERA-The French Aerospace Lab 7AD rotor in hover flight was computed. The tip Mach number was $M_{\text{tip}} = 0.6612$ and the collective pitch $\theta_{0.7} = 7.5^\circ$. A multi-block mesh with 880,000 cells per blade was used, and the effect of the other blades was accounted for using periodic boundary conditions (see figure 10). At the far-field boundaries, Froude conditions were imposed to better represent the flow induced by the rotor and to avoid the formation of artificial recirculation regions. The CFL number for the base flow and the tangent solvers was set to 8.

The derivatives of the thrust coefficient C_T and of the torque coefficient C_Q computed with HMB2 solving the sensitivity equation in tangent and adjoint mode are shown in table 15, where θ denotes the collective angle of the rotor, and θ_{tw} the blade twist angle centred at 70% of the blade span. Since no sensitivity result is available in the literature for this rotor, the sensitivities computed with the automatically differentiated code were compared with those obtained via second order centred finite differences. The comparison relative to the collective angle sensitivity is shown in table 16, where a good agreement between the two methods can be observed.

The method for computing the partial derivatives with respect to the collective angle is based on the chain rule expansion (26) and mesh deformation using IDW, differentiated with AD. The variation of the control angle is introduced as a differential deformation field which represents a rotation of the blade surface that vanishes at the hub, the far-field and the periodic planes. The method is general and allows to compute the flow derivative with respect to an arbitrary deformation of the blade, a feature particularly useful for the application of the adjoint method in aerodynamic shape optimization. For instance, figure 11 shows the

	C_L	C_D	C_Y	C_l	C_m	C_n
α	5.5156E+00	4.6532E-01	-3.2669E-11	2.4101E-10	-4.0997E+00	-9.6397E-12
β	-2.0897E-10	-2.1387E-11	-7.1952E-03	-1.2674E-01	2.1112E-10	1.6746E-02
M_∞	7.8775E-01	1.5873E-01	6.8438E-11	4.9994E-10	-9.0499E-01	-1.2981E-10
p	-3.7451E-10	-1.6635E-10	2.4530E-01	-1.5212E+00	4.5043E-10	-2.1644E-01
q	1.2773E+01	5.8386E-01	-6.6657E-11	7.0010E-10	-1.0789E+01	-3.5469E-11
r	1.2696E-09	1.9920E-10	-5.2381E-02	4.5353E-01	-1.5312E-09	2.5173E-02

HMB2 tangent mode

	C_L	C_D	C_Y	C_l	C_m	C_n
α	5.5772E+00	4.5422E-01	0.0000E+00	0.0000E+00	-4.0932E+00	0.0000E+00
β	-1.5168E-05	4.0961E-06	-6.8243E-03	-1.2667E-01	1.4722E-05	1.4088E-02
M_∞	7.7872E-01	1.3225E-01	0.0000E+00	0.0000E+00	-8.3126E-01	0.0000E+00
p	-2.2748E-06	3.3527E-07	2.3829E-01	-1.4971E+00	2.0630E-06	-2.1088E-01
q	1.3474E+01	6.0271E-01	0.0000E+00	0.0000E+00	-1.1437E+01	0.0000E+00
r	1.5217E-06	-4.4730E-07	-4.6747E-02	4.4085E-01	-1.4838E-06	2.3991E-02

Mader *et al.* [2]

	C_L	C_D	C_Y	C_l	C_m	C_n
α	1.105	2.443			0.159	
β			5.435	0.053		18.865
M_∞	1.160	21.024			8.869	
p			2.940	1.609		2.634
q	5.201	3.127			5.667	
r			12.052	2.877		4.927

Difference [%]

Table 14: Comparison of HMB2 sensitivity results with reference results for the M6 wing (inviscid flow, $\alpha = 3.06^\circ$, $M_\infty = 0.8395$).

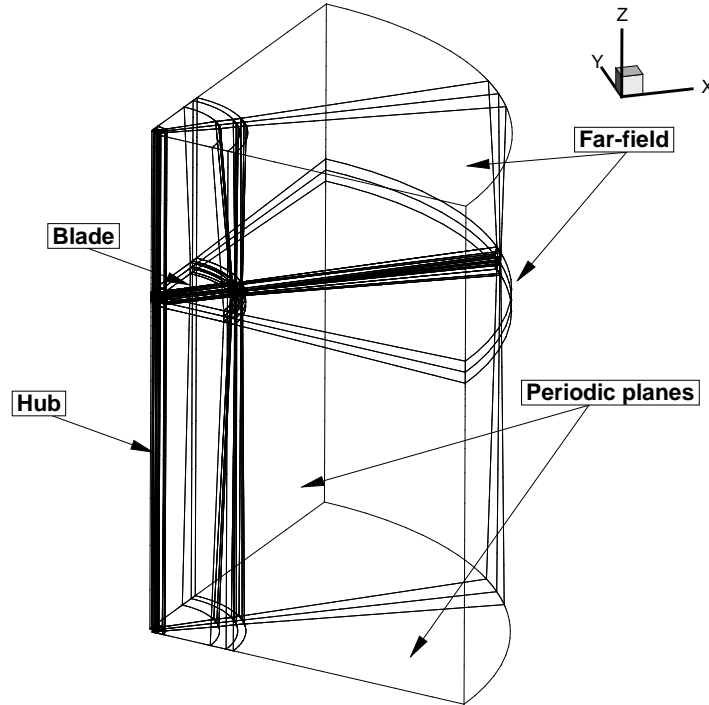


Figure 10: Mesh topology for the 7AD rotor in hover.

surface distribution of the pressure derivative with respect to a twist variation centred at the blade section with $r/R = 0.7$, while figure 12 shows the distribution of the adjoint variable of the thrust coefficient C_T relative to the pressure equation.

	C_T	C_Q
θ	0.1191865	0.0125512
θ_{tw}	-0.0044539	-0.0009800

Table 15: HMB2 sensitivity results for the ONERA-The French Aerospace Lab 7AD rotor (inviscid flow, $\theta_{0.7} = 7.5^\circ$, $M_{tip} = 0.6612$).

	$\partial C_T / \partial \theta$	$\partial C_Q / \partial \theta$
HMB2 AD tangent	0.1191865	0.0125512
HMB2 AD adjoint	0.1191865	0.0125512
HMB2 FD	0.1200453	0.0125287
Difference [%]	0.730	0.180

Table 16: Comparison of collective angle sensitivity computed with AD and FD for the ONERA-The French Aerospace Lab 7AD rotor (inviscid flow, $\theta_{0.7} = 7.5^\circ$, $M_{tip} = 0.6612$).

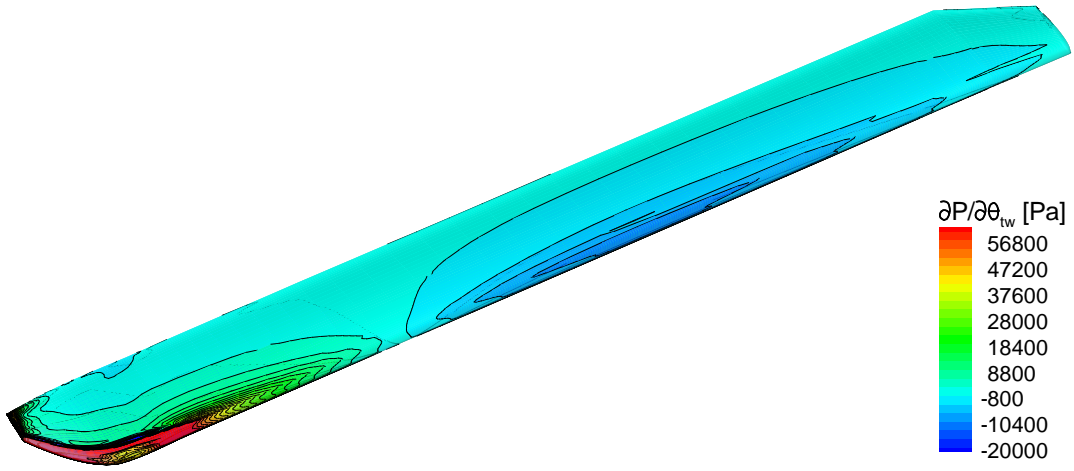


Figure 11: Surface pressure sensitivity to the twist angle (7AD, inviscid flow, $\theta_{0.7} = 7.5^\circ$, $M_{tip} = 0.6612$).

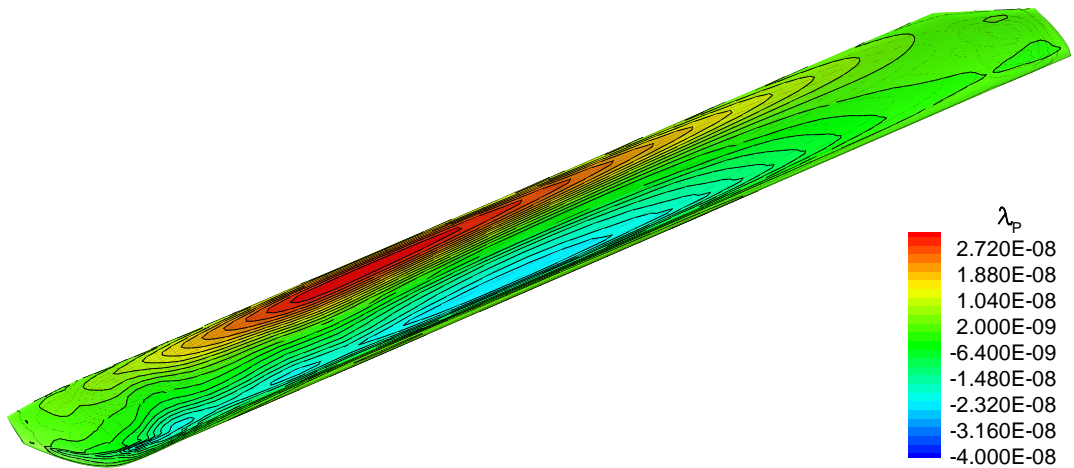


Figure 12: Pressure equation adjoint variable of the thrust coefficient (7AD, inviscid flow, $\theta_{0.7} = 7.5^\circ$, $M_{tip} = 0.6612$).

The computational time for the base flow of the rotor case was 6.3 hours on using 8 cores (3.3GHz Intel Xeon E31245). The time spent for each one of the three sensitivity solutions in tangent mode was 5 hours (80% of the base flow time).

8.8 Viscous Flow Around the S-76 Rotor in Hover

An accurate estimation of rotor flight-mechanic derivatives requires high-fidelity modelling of the flow. The sensitivity equation approach has been therefore extend to the full three-dimensional RANS equations completed with the $k-\omega$ turbulence model [51]. To demonstrate the method, the flow sensitivity of a S-76 rotor in hover flight [54] was analyzed, with viscosity and turbulence model. The tip Mach number was $M_{\text{tip}} = 0.65$, the Reynolds number was $\text{Re} = 1.1 \cdot 10^6$ and the collective pitch $\theta_{0.75} = 7.5^\circ$. The mesh had 9 million cells per blade, and the effect of the other blades was modelled with periodic boundary conditions. At the far-field boundaries, Froude conditions were imposed. The CFL number for the base flow solver and for the tangent solver was set to 3.

Note that the second order Jacobian of the flow equations for the S-76 mesh needs about 52GB of memory storage. It is clear that memory requirement poses a severe limit to the application of methods needing explicit storage of the Jacobian matrix. The matrix-free approach, on the other hand, alleviates this memory requirement and proves to be particularly suited for large-scale applications.

The derivatives of the thrust coefficient C_T and of the torque coefficient C_Q computed with HMB2 solving the sensitivity equation in tangent mode are shown in table 17. Fairly good agreement was found between the obtained sensitivity of the thrust coefficient with respect to the collective, and the theoretical value computed by:

$$(31) \quad \frac{\partial C_T}{\partial \theta} = \frac{\sigma a}{6} \left[1 - \frac{1}{\sqrt{1 + 64/(3\sigma a)\theta}} \right] = 0.0909,$$

In the above $\sigma = 0.0704$ and $a = 6.113$. Note that the Euler computation for the ONERA-The French Aerospace Lab 7AD case overestimates the derivative $\partial C_T / \partial \theta$, and that high-fidelity flow modelling leads to a better estimate. Table 18 shows a comparison between the collective angle sensitivity computed with AD and second order centred FD. The relative error is less than 0.6% for both the thrust and torque coefficients.

	C_T	C_Q
θ	0.0922836	0.0090250
θ_{tw}	-0.0035866	-0.0006710

Table 17: HMB2 sensitivity results for the S-76 rotor ($k-\omega$ turbulence model, $\theta_{0.75} = 7.5^\circ$, $M_{\text{tip}} = 0.65$, $\text{Re} = 1.1 \cdot 10^6$).

	$\partial C_T / \partial \theta$	$\partial C_Q / \partial \theta$
HMB2 AD tangent	0.0922836	0.0090250
HMB2 FD	0.0917879	0.0089743
Difference [%]	0.537	0.562

Table 18: Comparison of collective angle sensitivity computed with AD and FD for the S-76 rotor ($k-\omega$ turbulence model, $\theta_{0.75} = 7.5^\circ$, $M_{\text{tip}} = 0.65$, $\text{Re} = 1.1 \cdot 10^6$).

9 CONCLUSIONS

This paper presented the implementation and assessment of AD and discrete adjoint methods within the framework of implicit CFD solvers. The method benefits from the fully implicit formulation of the adjoint solver, and achieves a low memory footprint by avoiding the storage of the high-order Jacobian. After implementation, validation of the method has been attempted using established cases found in the literature for airfoils and wings. The results show that the current method achieves results in agreement with theory and with published solutions. The aerodynamic derivatives for rotors in hover were also computed and results were found to agree with finite difference computations. To cope with periodic solutions, such as the flow of rotors in forward flight, the sensitivity equation approach was applied to the harmonic balance formulation of the RANS equations. The method was validated with pitching airfoil cases in inviscid and turbulent flows. The cost for solving the sensitivity equation with the fully implicit method is usually lower than that of the nonlinear base flow solution, even for complex flow cases. In the future the method will be used alongside a gradient based optimization method for studies of rotors in hover and forward flight.

Acknowledgement The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement n° 264672.

The differentiated computer code can be made available to interested readers.

REFERENCES

- [1] A. C. Limache and E. M. Cliff. “Aerodynamic Sensitivity Theory for Rotary Stability Derivatives”. In: *Journal of Aircraft* 37.4 (2000), pp. 676–683. DOI: 10.2514/2.2651.
- [2] C. A. Mader and J. R. R. A. Martins. “Computation of Aircraft Stability Derivatives Using an Automatic Differentiation Adjoint Approach”. In: *AIAA Journal* 49.12 (2011), pp. 2737–2750. DOI: 10.2514/1.J051147.
- [3] M. B. Giles and N. A. Pierce. “An Introduction to the Adjoint Approach to Design”. In: *Flow, Turbulence and Combustion* 65.3-4 (2000), pp. 393–415. DOI: 10.1023/A:1011430410075.
- [4] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. 2nd ed. Philadelphia, USA: Society for Industrial and Applied Mathematics, 2008, pp. 245–258. DOI: 10.1137/1.9780898717761.
- [5] A. Griewank. “A Mathematical View of Automatic Differentiation”. In: *Acta Numerica*. Vol. 12. Cambridge University Press, May 2003, pp. 321–398. DOI: 10.1017/S0962492902000132.
- [6] M. C. Bartholomew-Biggs et al. “Automatic Differentiation of Algorithms”. In: *Journal of Computational and Applied Mathematics* 124 (2000), pp. 171–190. DOI: 10.1016/S0377-0427(00)00422-2.
- [7] R. Giering, T. Kaminski, and T. Slawig. “Generating Efficient Derivative Code with TAF: Adjoint and Tangent Linear Euler Flow Around an Airfoil”. In: *Future Generation Computer Systems* 21.8 (2005), pp. 1345–1355. ISSN: 0167-739X. DOI: <http://dx.doi.org/10.1016/j.future.2004.11.003>.
- [8] C. A. Mader et al. “ADjoint: An Approach for the Rapid Development of Discrete Adjoint Solvers”. In: *AIAA Journal* 46.4 (2008), pp. 863–873. DOI: 10.2514/1.29123.
- [9] D. Jones, J.-D. Müller, and F. Christakopoulos. “Preparation and Assembly of Discrete Adjoint CFD Codes”. In: *Computers and Fluids* 46.1 (2011), pp. 282–286. DOI: 10.1016/j.compfluid.2011.01.042.
- [10] R. Steijl, G. N. Barakos, and K. Badcock. “A Framework for CFD Analysis of Helicopter Rotors in Hover and Forward Flight”. In: *International Journal for Numerical Methods in Fluids* 51.8 (2006), pp. 819–847. DOI: doi:10.1002/(ISSN)1097-0363.
- [11] G. Barakos et al. “Development of CFD Capability for Full Helicopter Engineering Analysis”. In: *31st European Rotorcraft Forum* [CD-ROM]. Paper 91. Council of European Aerospace Societies, 2005.
- [12] O. Pironneau. “On Optimum Design in Fluid Mechanics”. In: *Journal of Fluid Mechanics* 64.1 (1974), pp. 97–110.
- [13] A. Jameson. “Aerodynamic Design Via Control Theory”. In: *Journal of Scientific Computing* 3.3 (1988), pp. 233–260. DOI: 10.1007/BF01061285.
- [14] A. Jameson, L. Martinelli, and N. A. Pierce. “Optimum Aerodynamic Design Using the Navier-Stokes Equations”. In: *Theoretical and Computational Fluid Dynamics* 10.1–4 (1998), pp. 213–237.
- [15] A. Jameson. “Control Theory for Optimum Design of Aerodynamic Shapes”. In: *Proceedings of the IEEE Conference on Decision and Control*. Vol. 1. New York: Inst. of Electrical and Electronics Engineers, 1990, pp. 176–179.
- [16] A. Jameson, N. A. Pierce, and L. Martinelli. “Optimum Aerodynamic Design Using the Navier-Stokes Equations”. In: *35th AIAA Aerospace Sciences Meeting and Exhibit* AIAA Paper 97-0101 (1997), pp. 1–22.
- [17] J. Reuther and A. Jameson. “Control Theory Based Airfoil Design Using the Euler Equations”. In: *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization* 94.AIAA Paper 1994-CP4272 (1994), pp. 1–17.
- [18] J. Reuther et al. “Aerodynamic Shape Optimization of Complex Aircraft Configurations Via an Adjoint Formulation”. In: *34th AIAA Aerospace Sciences Meeting and Exhibit* AIAA Paper 1996-0094 (1996). Also AIAA Paper 1996-0094, 1996.
- [19] J. Elliott and J. Peraire. “Practical Three-Dimensional Aerodynamic Design and Optimization Using Unstructured Meshes”. In: *AIAA Journal* 35.9 (1997), pp. 1479–1485. DOI: 10.2514/2.271.
- [20] W. K. Anderson and D. L. Bonhaus. “Airfoil Design on Unstructured Grids for Turbulent Flows”. In: *AIAA Journal* 37.2 (1999), pp. 185–191. DOI: 10.2514/2.712.
- [21] D. Mavriplis. “Discrete Adjoint-Based Approach for Optimization Problems on Three-Dimensional Unstructured Meshes”. In: *AIAA Journal* 45.4 (2007), pp. 740–750. DOI: 10.2514/1.22743.
- [22] J. C. Newman III et al. “Overview of Sensitivity Analysis and Shape Optimization for Complex Aerodynamic Configurations”. In: *Journal of Aircraft* 36.1 (1999), pp. 87–96. DOI: 10.2514/2.2416.
- [23] B. Mohammadi. “Optimal Shape Design, Reverse Mode of Automatic Differentiation and Turbulence”. In: *35th Aerospace Sciences Meeting and Exhibit* AIAA Paper 97-0099 (Jan. 1997).

- [24] B. Mohammadi. “Practical Application to Fluid Flows of Automatic Differentiation for Design Problems”. In: *Inverse Design and Optimization Methods, Lecture Series 1997-05*. Rhode Saint Genese, Belgium: von Karman Institute for Fluid Dynamics, Apr. 1997, pp. M1–M34.
- [25] J.-D. Müller and P. Cusdin. “On the Performance of Discrete Adjoint CFD Codes Using Automatic Differentiation”. In: *International Journal for Numerical Methods in Fluids* 47.8-9 (2005), pp. 939–945. DOI: 10.1002/flid.885.
- [26] A. C. Marta et al. “A Methodology for the Development of Discrete Adjoint Solvers Using Automatic Differentiation Tools”. In: *International Journal of Computational Fluid Dynamics* 21.9-10 (2007), pp. 307–327. DOI: 10.1080/10618560701678647.
- [27] A. Carle and M. Fagan. “ADIFOR 3.0 Overview”. In: *Rice Univ., TR CAAM-TR-00-02* (2000).
- [28] R. Giering, T. Kaminski, and T. Slawig. “Generating Efficient Derivative Code With TAF: Adjoint and Tangent Linear Euler Flow Around an Airfoil”. In: *Future Generation Computer Systems* 21.8 (2005), pp. 1345–1355. DOI: doi:10.1016/j.future.2004.11.003.
- [29] M. S. Gockenbach. *Understanding Code Generated by TAMC*. IAAA Paper TR00-29. Houston, TX: Department of Computational and Applied Mathematics, Rice Univ., 2000.
- [30] L. Hascoët and V. Pascual. *TAPENADE 2.1 User’s Guide*. Tech. rep. TR 300. Sophia Antipolisi, France: Institut National de Recherche en Informatique et en Automatique, 2004.
- [31] Anon. *The TAPENADE Tutorial*. <http://www-sop.inria.fr/tropics/tapenade.html> [retrieved 1 May 2015].
- [32] Anon. *Community Portal for Automatic Differentiation*. <http://www.autodiff.org> [retrieved 1 May 2015].
- [33] C. A. Mader and J. R. R. A. Martins. “Derivatives for Time-Spectral Computational Fluid Dynamics Using an Automatic Differentiation Adjoint”. In: *AIAA Journal* 50.12 (2012), pp. 2809–2819. DOI: 10.2514/1.J051147.
- [34] S. Choi et al. “Helicopter Rotor Design Using a Time-Spectral and Adjoint-Based Method”. In: *Journal of Aircraft* 51.2 (2014), pp. 412–423. DOI: 10.2514/1.C031975.
- [35] H. Huang and K. Ekici. “A Discrete Adjoint Harmonic Balance Method for Turbomachinery Shape Optimization”. In: *Aerospace Science and Technology* 39 (2014), pp. 481–490. DOI: 10.1016/j.ast.2014.05.015.
- [36] R. P. Dwight and J. Brezillon. “Effect of Approximations of the Discrete Adjoint on Gradient-Based Optimization”. In: *AIAA Journal* 44.12 (2006), pp. 3022–3031. DOI: 10.2514/1.21744.
- [37] Z. Lyu et al. “Automatic Differentiation Adjoint of the Reynolds-Averaged Navier-Stokes Equations with a Turbulence Model”. In: *21st AIAA Computational Fluid Dynamics Conference*. San Diego, CA, June 2013. DOI: 10.2514/6.2013-2581.
- [38] A. C. Marta et al. “Interpretation of Adjoint Solutions for Turbomachinery Flows”. In: *AIAA Journal* 51.7 (2013), pp. 1733–1744. DOI: 10.2514/1.J052177.
- [39] J. P. Thomas, E. H. Dowell, and K. C. Hall. “Discrete Adjoint Method for Nonlinear Aeroelastic Sensitivities for Compressible and Viscous Flows”. In: *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. Boston, Apr. 2013, pp. 6144–6152. DOI: 10.2514/6.2013-1860.
- [40] L. Osusky and D. W. Zingg. “Application of an Efficient Newton-Krylov Algorithm for Aerodynamic Shape Optimization Based on the Reynolds-Averaged Navier-Stokes Equations”. In: *21st AIAA Computational Fluid Dynamics Conference*. AIAA Paper 2013-2584. June 2013, pp. 1–18. DOI: 10.2514/2013-2584.
- [41] Z. Lyu and J. R. R. A. Martins. “Aerodynamic Design Optimization Studies of a Blended-Wing-Body Aircraft”. In: *Journal of Aircraft* 51.5 (2014), pp. 1604–1617. DOI: 10.2514/1.C032491.
- [42] K. Mani and D. J. Mavriplis. “Adjoint-Based Sensitivity Formulation for Fully Coupled Unsteady Aeroelasticity Problems”. In: *AIAA Journal* 47.8 (2009), pp. 1902–1915. DOI: 10.2514/1.40582.
- [43] K. C. Hall et al. “Harmonic Balance Methods Applied to Computational Fluid Dynamics Problems”. In: *International Journal of Computational Fluid Dynamics* 27.2 (2013), pp. 52–67. DOI: 10.1080/10618562.2012.742512.
- [44] M. Woodgate and G. Barakos. “Implicit Computational Fluid Dynamics Methods for Fast Analysis of Rotor Flows”. In: *AIAA Journal* 50.6 (2012), pp. 1217–1244. DOI: doi:10.2514/1.J051155.
- [45] A. Jameson. “Time Dependent Calculations Using Multigrid with Application to Unsteady Flows past Airfoils and Wings”. In: *10th Computational Fluid Dynamics Conference* AIAA Paper 1991-1596 (June 1991).
- [46] O. Axelsson. *Iterative Solution Methods*. Cambridge, MA: Cambridge Univ. Press, 1994, pp. 504–557.
- [47] B. Christianson. “Reverse Accumulation and Implicit Functions”. In: *Optimization Methods and Software* 9.4 (1998), pp. 307–322. DOI: doi:10.1080/10556789808805697.

- [48] M. B. Giles. “On the Iterative Solution of Adjoint Equations”. In: *Automatic Differentiation of Algorithms*. Ed. by G. Corliss et al. Springer New York, 2002, pp. 145–151.
- [49] D. Shepard. “A Two-dimensional Interpolation Function for Irregularly-spaced Data”. In: *Proceedings of the 1968 23rd ACM National Conference*. New York: Assoc. for Computing Machinery, 1968, pp. 517–524.
- [50] D.-H. Kim, J.-W. Chang, and J. Chung. “Low-Reynolds-Number Effect on Aerodynamic Characteristics of a NACA 0012 Airfoil”. In: *Journal of Aircraft* 48.4 (2011), pp. 1212–1215. DOI: 10.2514/1.C031223.
- [51] D. C. Wilcox. “Re-Assessment of the Scale-Determining Equation for Advanced Turbulence Models”. In: *AIAA Journal* 26.11 (1988), pp. 1299–1310. DOI: 10.2514/3.10041.
- [52] Anon. *Compendium of Unsteady Aerodynamic Measurements*. Tech. rep. AGARD TR 702. Neuilly-sur-Seine, France: NATO Science and Technology Organization, Aug. 1982.
- [53] Anon. *Verification and Validation Data for Computational Unsteady Aerodynamics*. Tech. rep. RTO-TR-26. Neuilly-sur-Seine, France: NATO Science and Technology Organization, Oct. 2000.
- [54] D. T. Balch and J. Lombardi. *Experimental Study of Main Rotor Tip Geometry and Tail Rotor Interactions in Hover. Volume I. Text and Figures*. NASA CR 177336. NASA, Feb. 1985.